T.C.
MİLLÎ EĞİTİM BAKANLIĞI

ENDÜSTRİYEL OTOMASYON TEKNOLOJİLERİ

PROGRAMLAMA UYGULAMALARI

Ankara, 2015

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- PARA İLE SATILMAZ.

İÇİNDEKİLER

AÇIKLAMALAR ii
GİRİŞ 1
ÖĞRENME FAALİYETİ -1 3
1. DİZİLER 3
1.1. Tek Boyutlu Diziler 3
1.1.1. Dizinin Başlatılması 10
1.1.2. Sıralama 11
1.2. İki Boyutlu Dizi 15
UYGULAMA FAALİYETİ 18
ÖLÇME VE DEĞERLENDİRME 20
ÖĞRENME FAALİYETİ- 2 21
2. KARAKTER KATARLARI 21
2.1. Karakter ve Karakter Kodu 21
2.2. Karakter Katarları 25
2.3. Çok Yönlü Karar Yapısı 30
2.4. Çoklu Karar Yapısı 33
UYGULAMA FAALİYETİ 36
ÖLÇME VE DEĞERLENDİRME 39
ÖĞRENME FAALİYETİ- 3 41
3. DÖNGÜLER 41
3.1. While Döngüsü 41
3.2. For Döngüsü 48
3.2.1. İç İçe Döngüler 51
UYGULAMA FAALİYETİ 54
ÖLÇME VE DEĞERLENDİRME 56
MODÜL DEĞERLENDİRME 58
CEVAP ANAHTARLARI 59
KAYNAKÇA 60

AÇIKLAMALAR

ALAN	**Endüstriyel Otomasyon Teknolojileri**
DAL/MESLEK	**Ortak Alan**
MODÜLÜN ADI	**Programlama Uygulamaları**
MODÜLÜN TANIMI	Bilgisayar programı yazım tekniklerini anlatan öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	
YETERLİK	Modül tamamlandığında bilgisayar programı yapmak
MODÜLÜN AMACI	**Genel Amaç** Bilgisayar programını programlama dili tekniklerine uygun olarak yazabileceksiniz. **Amaçlar** **1.** Bilgisayar programlamada dizi yapısındaki verileri doğru bir şekilde kullanabileceksiniz. **2.** Bilgisayar programlamada karakter ve karakter katarlarını doğru bir şekilde kullanabileceksiniz. **3.** Bilgisayar programlamada disk dosyalarının yazma, okuma ve ekleme işlemlerini doğru bir şekilde yapabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Donanım:** Bilgisayar, bilgisayar çevre birimleri, programlama akış şablonu **Ortam:** Programlama laboratuvarı
ÖLÇME VE DEĞERLENDİRME	Modülün içinde yer alan, her faaliyetten sonra verilen ölçme araçları ile kazandığınız bilgileri ölçerek kendi kendinizi değerlendireceksiniz. Öğretmen, modülün sonunda, size ölçme aracı (test, çoktan seçmeli, doğru-yanlış, vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek değerlendirecektir.

GİRİŞ

Sevgili Öğrenci,

Bundan önceki modülde C dilinin temelleri üzerinde çalıştınız. Bu bilgiler vazgeçilmez olgulardır ve sınırlı kullanıma sahip programlar için yeterlidir. Başka bir ifadeyle bu temel bilgiler fazla veriye sahip daha büyük programlar için yeterli değildir. Program dışında veri taşıyabilen yardımcı dosyalar ve program yapısını oluşturma imkânı veren fonksiyonlarla, daha karışık ve uygulamaya yönelik programlar yazabiliriz. Bu nedenle modülde sistem gelişim süreci ile ilişkili olan bu konular üzerinde çalışacaksınız.

Programlar 50 satırı aştığı zaman, iyi bir yapıyla net bir şekilde düzenlenmeleri zorunlu hale gelir. Aksi takdirde programın ayıklanması, çok fazla zaman alacak ve programcı sonradan bazı bölümleri değiştirmek istediği zaman çok fazla zorluklarla yüz yüze gelecektir. Bu bilgiler bireysel tecrübelerden elde edilmiştir. Bu modülde kendinizi, kendi fikirlerinizden yola çıkarak büyük programlar yapmaya teşvik etmelisiniz.

ÖĞRENME FAALİYETİ -1

AMAÇ

Bilgisayar programlamada dizi yapısındaki verileri doğru bir şekilde kullanabileceksiniz.

ARAŞTIRMA

- Sevgili öğrenci, bu öğrenme faaliyetinden önce aşağıdaki hazırlıkları yapınız.
- C programlama dili kuralları hakkında araştırma yapınız.

1. DİZİLER

C programlama dilinde veriler değişken yapısı içinde saklanır. Bir değişken sadece bir veriyi tutar. Programda bir çok veriyi kullanmak istendiğinde ise dizi adı verilen veri yapıları kullanılır. Dizi, belirli sayıda verinin bellekte saklandığı değişken listeleridir.

1.1. Tek Boyutlu Diziler

Aşağıdaki program C dilinde en temel yapıya bir örnek olarak gösterilebilir.

Örnek:

4 adet integer tipindeki sayıyı tutan ve bunları giriş sırasına göre görüntüleyen programı yazınız.

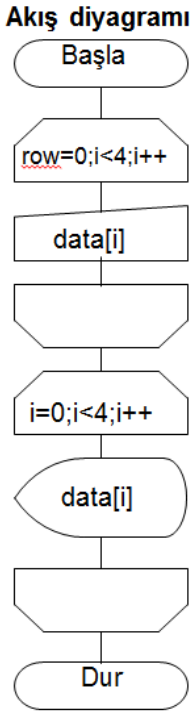
Ekran görüntüsü

```
Sayı Giriniz =>34
Sayı Giriniz  =>41
Sayı Giriniz  =>97
Sayı Giriniz  =>53
34
41
97
53
```

Değişken tablosu

Değişken	tip	Kullanım
data[4]	int	Saklanan sayı
i	int	Kontrol değişkeni

Şekil 1.1 : Ekran görüntüsü ve değişken tablosu



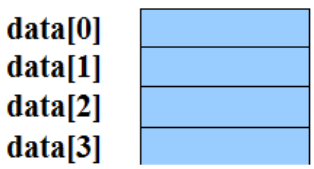

```
/* Örnek 1.1 */

#include <stdio.h>

main()
{
     int data[4],i;                          //--1

    for(i=0;i<4;i++)
   {                                         //--2
        printf("Sayı giriniz=>");
        scanf("%d",&data[i]);
    }
   for(i=0;i<4;i++)                          //--3
        printf("%d \n",data[i]);
}
```

Şekil 1.2 : Akış diyagramı ve program

- Program açıklaması

  - int data[4]; komutuyla dizinin tanımlanması,
  - 1. maddede tanımlanan dizi için klavyeden bilgi girişinin yapılması,
  - Dizinin içeriğinin görüntülenmesidir.

int data[4] şeklindeki bir tanımlamayla, şekilde görüldüğü gibi, ana bellekte sıralı olarak hafıza alanı hazırlanır. Bu yapıya dizi adı verilir. Dizilerin kullanım yapısı şu şekildedir.



Şekil 1.3 : Data haritası

Dizinin her elemanına bir dizi ismi ve köşeli parantezlerin içerisine yazılan eleman numarası ile ulaşabiliriz. Dizilerde köşeli parantezler (subscript) için 0 dan başlama kuralı olduğundan, **data[4]** şeklindeki bir tanımlamada, 4 eleman **data[0]**, **data[1]**, **data[2]**, **data[3]** ü ifade eder.

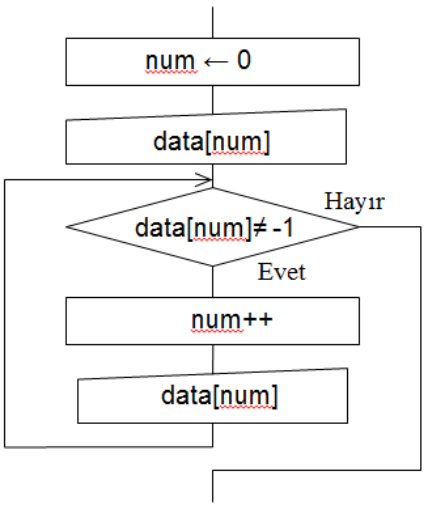
Örneğin, **data[1]**=5 yazdığımızda dizinin ikinci elemanına sayısal 5 değerini aktarıyoruz demektir.

Aynı zamanda köşeli parantezler içerisine değişken ismini de yazabiliriz.

```
i=1;

data[i]=5;
```

Yukarıdaki ifade bir öncekiyle aynıdır. Bu yazım şeklinin döngü içinde kullanılması diziler üzerinde işlem yapabilmemizi sağlar.

Örnekte veri sayısı 4 ile sınırlıdır. Programı istediğimiz kadar veriyi kullanabilmek için değiştirebiliriz. Bunu gerçekleştirebilmek için gerekli sayıda eleman kullanan bir dizi aşağıdaki örnekte tanıtılmıştır.



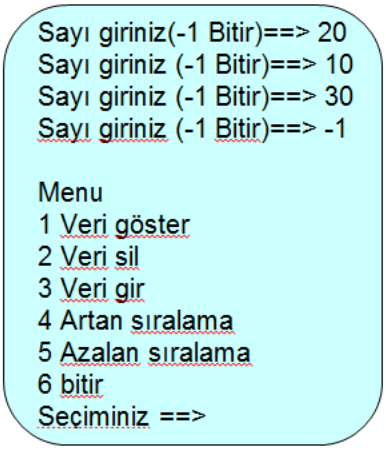
Bu programda dizi sonu - 1 değerinin girilmesiyle kontrol edilmektedir.
Verilerin girilmesinden sonra num değişkeni veri sayısını tutmaktadır.

Şekil 1.4 : Karar yapılı döngü kontrolü

Örnek:

İsteğe bağlı olarak 10`a kadar sayı girilebilen ve menüde gösterilen işlemleri gerçekleştirecek programı yazınız. Veri ekleme ve sıralama daha sonra anlatılacaktır.

Ekran görüntüsü

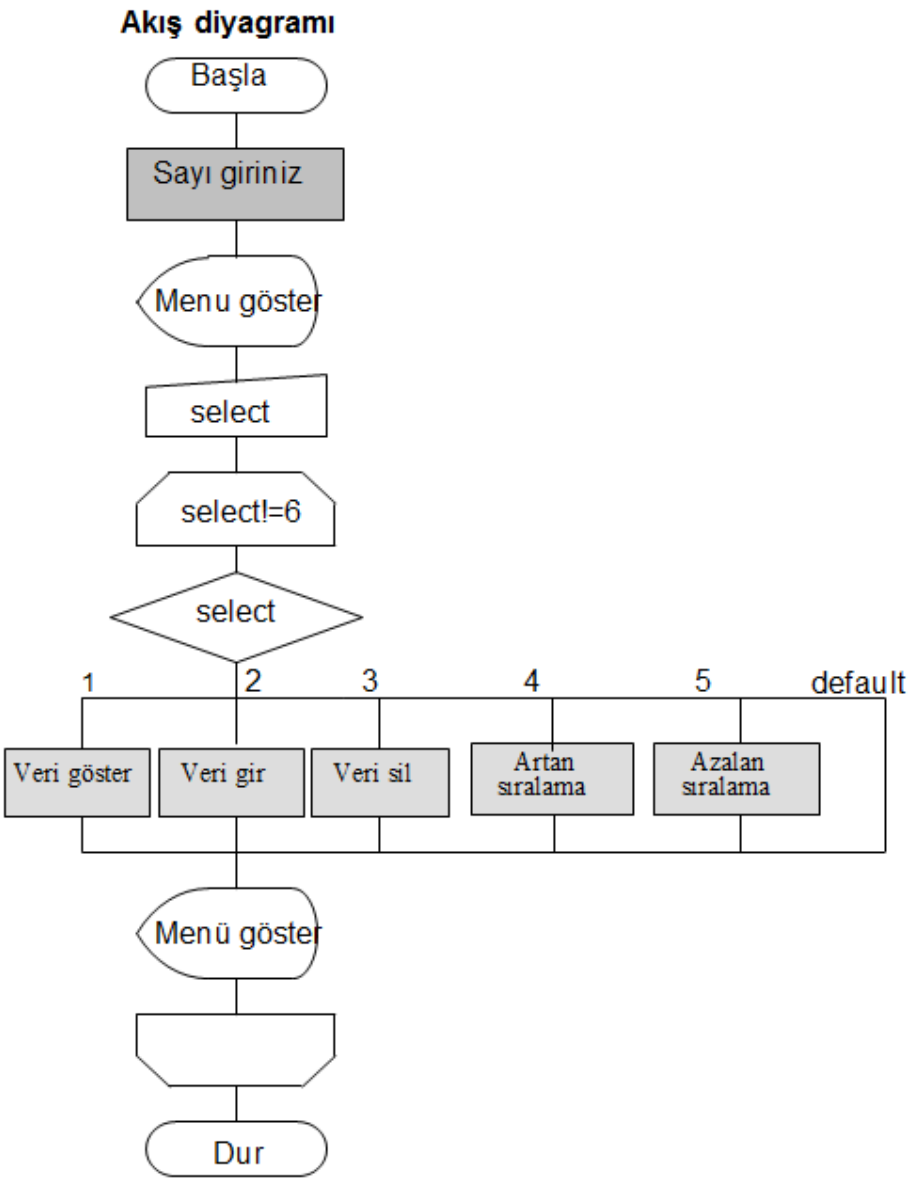


Değişken tablosu

değişken	tip	Kullanım
data[11]	int	Sayıyı saklama
select	int	Menü seçimi
num	int	Veri sayısı
i	int	Kontrol değişkeni

Şekil 1.5 : Ekran görüntüsü ve değişken tablosu

Bu programda karmaşıklığa sebep olmamak için birçok printf komutu menü fonksiyonunda kullanılmıştır. Buna göre programa veri girildiğinde menü fonksiyonu işletilecektir. Bu fonksiyonu bir alt program olarak düşünebiliriz. Başka bir ifadeyle programın herhangi bir yerinde **menu();** yazarak printf komutlarının grup halinde yazıldığı menü fonksiyonunu çağırabiliriz. C programlamada fonksiyon isimlerine programcı karar verir.



Şekil 1.6: Akış diyagramı

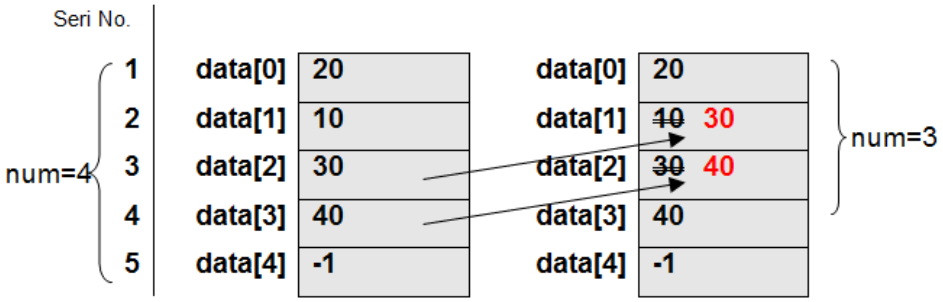
Burada genel bir akış diyagramı program yapısının tamamını göstermek için kullanılmaktadır. Kullanılan kutular tek bir süreci değil bir grup çalışmayı gösterir. Bu şekilde büyük programlar yaptığınızda, öncelikle programın genel yapısını tasarlayıp sonra detaya girmek istenilen bir durumdur.

Program:

```
/* Ornek 1.2 */

#include<stdio.h>
main()
{
    int data[11],select,num,i;

    num = 0;
    printf("Sayı giriniz(-1 bitir)==>");
    scanf("%d",&data[num]);
    while (data[num] != -1){
          num++;
          printf("Sayı giriniz(-1 bitir)==>");
          scanf("%d",&data[num]);
    }
    menu();
    scanf("%d",&select);
    while( select != 6) {
        switch (select){
           case 1:
                printf("\nNo.  Number\n");
                 i=0;
                while(i<num) {
                    printf("%2d %6d\n",i+1,data[i]);
                    i++;
                }
                break;
            case 2:
      printf("Silmek istediğiniz numarayı giriniz ==> ");
                scanf("\d",&i);
                while (i < num){
                    data[i-1]=data[i];
                    i++;
                }
                        num--;
                break;
           case 3:
               /*  veri girişi için program */
                break;
           case 4:
              /*  artan sıralama için program */
                break;
           case 5:
               /* azalan sıralama için program */
                break;
        }
        menu();
        scanf("%d",&select);
    }
}
menu()
{
    printf("\nMenu\n");
    printf("1 – Veri göster\n");
    printf("2 – Veri sil\n");
    printf("3 – Veri gir\n");
    printf("4 – Artan sıralama\n");
    printf("5 – Azalan sıralama\n");
    printf("6 -- Bitir\n");
    printf("Seçiminiz ==> ");
}
```

➢ **Program açıklaması**

Verinin şekilde görüldüğü gibi dizi içindeki yerinin değiştirilmesi ile bu işlem gerçekleştirilir. Aşağıdaki şekilde dizi içindeki 10 sayısı siliniyor.



Şekil 1.7 : Silme işlemi

Yukarıda görülen şekli kullanarak, sürecin taslağını gösteren aşağıdaki gibi bir akış diyagramı çizebilmeliyiz. Bunun için (1), (2) ve (3) numaralı adımlara aşağıdaki gibi karar verilmiştir.

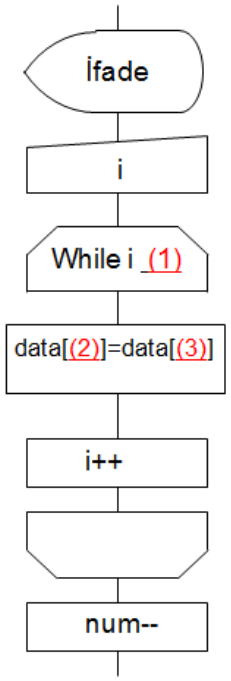
Şekil 1.7 göz önüne alındığında veriler şu şekilde taşınmıştır;

```
data[1]  ← data[2]
data[2]  ← data[3]
```

i değeri 2 den başladı. (2) ve (3) e şu şekilde karar verilir.

```
data[i-1]= data[i]
   (2)       (3)
```

Benzer şekilde, sayıları karşılaştırarak (1) değerini **< num** şeklinde sonlandırabiliriz.
-1 değerinin **data[4]'** ten **data[3]'** e taşınmasına ihtiyaç duymadığımız için, **num** değerinden 1 sayısını çıkarmak zorundayız. Bu aşamaları program takibinde dikkate almalısınız.



Şekil 1.8 : Akış diyagramı

1.1.1. Dizinin Başlatılması

Bir dizinin başlatılması işlemi şu şekilde tanımlanabilir,

```
int data[4] = { 456, 23, 567, 89};
```

Data[0]	456
Data[1]	23
Data[2]	567
Data[3]	89

Bu işlem dizinin tanımlanması ve ilk değerinin verilmesi ile tamamlanır. Dizinin başlatılması bir atama olmadığından aşağıdaki gibi yazılmalıdır.

```
int data[4];
data[4] = { 456, 23, 567, 89 };
```

Bazen başlatma yapılırken dizinin boyutunu ihmal edebiliriz.

```
int data[ ] = { 11,8,1};
```

Data[0]	11
Data[1]	8
Data[3]	1

Bu durumda dizinin boyutu yukarıdaki şekilde olduğu gibi otomatik olarak girilen veri adedi kadar ayarlanır.

1.1.2. Sıralama

Örnek:

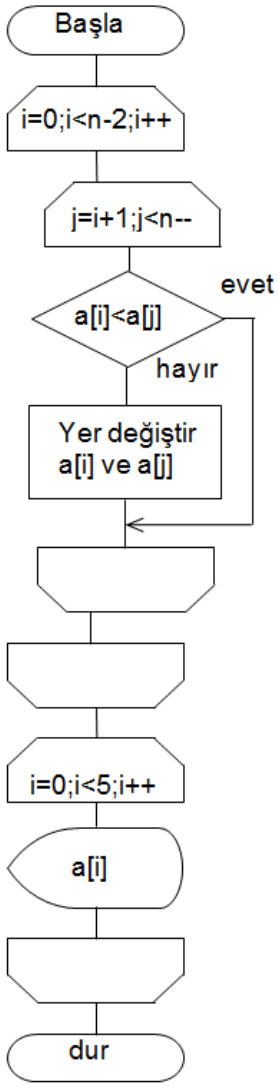
Bir dizideki sayıları azalan şekilde sıralayan ve görüntüleyen programı yazınız.

Ekran görüntüsü

Değişken tablosu

Değişken	tip	Kullanım
a[10]	int	Değerler
i	int	Kontrol değişkeni
j	int	Kontrol değişkeni

Akış diyagramı



Şekil 1.9 : Ekran görüntüsü, değişken tablosu ve akış diyagramı

Program:

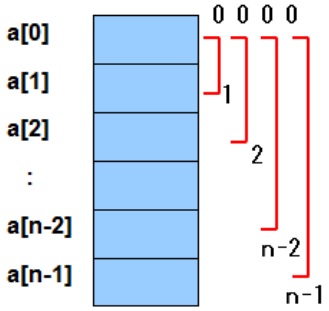
```
/* Ornek 1.3 */
#include <stdio.h>

main()
{
        int  n = 10;
        int  a[10]={3,4,3,2,8,4,6,7,8,9};
        int  i,j,work;

        for (i=0;i<n-1;i++)
        {
                for (j=i+1;j<n;j++)
                {
                        if(a[i] < a[j])
                        {
                                work=a[i];
                                a[i]=a[j];
                                a[j]=work;
                        }
                }
        }
        for(i=0;i<n;i++)
        printf("%d\n",a[i]);
}
```

- **Program açıklaması**

Sıralama aşağıdaki şekilde görüldüğü gibi yer değiştirerek gerçekleştirilir. Sıralama için birçok yol vardır. Biz yandaki metodu kullanacağız.

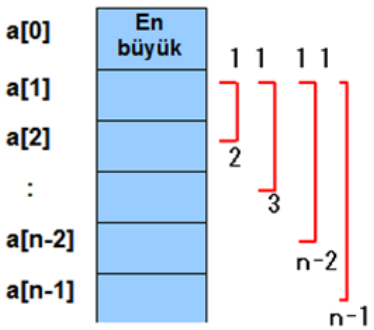
Artan sıralama	Azalan sıralama
1	15
2	8
4	6
6	4
8	2
15	1

1. adım (En büyük değerin **a[0]** a yerleştirilmesi)



İlk olarak, **a[0] ve a[1]** karşılaştırılır. **a[1]**' in büyük olduğu durumda ikisinin değeri yer değiştirilir. İkinci olarak **a[0]** ve **a[2]'** yi karşılaştırılır. **a[2] > a[0]** ise onların değeri yer değiştirilir. Karşılaştırmalar **a[0]'** dan **a[n-1]** değeri olan son elemana kadar devam edecektir. Son olarak en büyük değer **a[0]** içerisinde yer alacaktır.

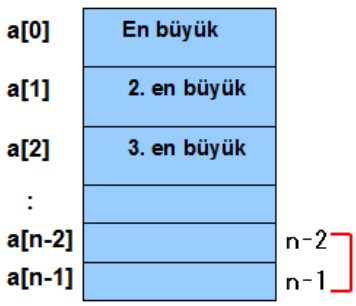
2. adım (En büyük değerin **a[1]** a yerleştirilmesi)



İkinci aşamada, **a[1]** ve **a[0]** haricindeki diğer bütün değerlerle karşılaştırmalar yapılır. Bu değerler **a[2]** den **a[n-1]'** e kadar olan değerlerdir. Bu şekilde, **a[1]** ikinci en büyük sayıya sahip olan alan olabilir.

Üçüncü aşama ve dördüncü aşama aynı durumdadır.

3.adım



Son aşama **a[n-2]** değeri için gerçekleştirilmelidir. Bu değer ve **a[n-1]** değeri için karşılaştırma ve yer değiştirme işlemleri anlaşılabilir**.**

Yukarıdaki basamaklar dış döngü (ilk döngü) yardımıyla gerçekleştirilir. Bu döngüde, kontrol değişkeni **i**, **0** dan **(a[0]**'ın ilk adımı için) **n-2 (a[n-2]** için son basamak) kadar değiştirilebilmelidir. Her adımdaki karşılaştırmalar 2. döngü ve iç döngü ile gerçekleştirilir. Kontrol değişkeni **j** burada, **i+1** den **n-1'** e kadar değiştirilebilmelidir. Tüm bu durumları bütün basamaklar için kontrol etmelisiniz.

a[i] ve **a[j]** arasındaki değerlerin yerlerini şu şekilde değiştiremeyiz**.**

```
a[i]=a[j];
a[j]=a[i];
```

Bu şekilde, a[i] değeri kaybolacaktır. a[i]'nin değerini geçici olarak saklayabilmek için şu ifadeleri kullanmalıyız.

```
   k=a[i];
a[i]=a[j];
a[j]=k;
```

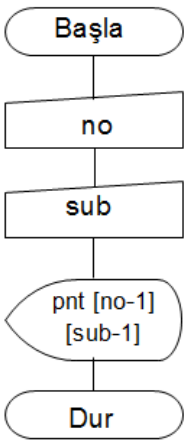
1.2. İki Boyutlu Dizi

Örnek: Sınav sonuçlarını iki boyutlu bir diziden okuyan programı yazınız.

Ekran görüntüsü

```
Öğrenci numarası ==>2
Hangi Dersin Notu ?
1--Matematik  2--Tarih  3--İngilizce
  ====>2
Notu  95.
```

Akış diyagramı



Değişken tablosu

Değişken	Tür	Kullanım
pnt[30][3]	int	Sınav Notu pnt[no][sub]
no	int	Öğrenci numarası 1—Matematik 2—Tarih 3—Ingilizce
sub	int	Ders numarası

Şekil 1.10 : Ekran görüntüsü, değişken tablosu ve akış diyagramı

Not: Bundan sonra, akış diyagramlarındaki bazı ifadeleri ihmal edeceğiz. Bu şekilde program yapısı daha açık hale gelecektir.

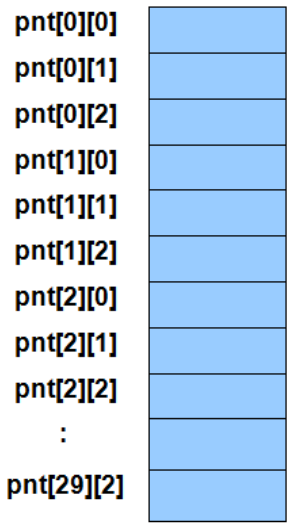
Program:

```
/* Ornek 1.4 */

#include <stdio.h>

main()
{
        int pnt[30][5]={{80,90,100},                    //--1
            {80,75,60},
              :
            {60,100,65}};
        int   no, sub;

        printf("Öğrenci numarasını giriniz ==>");
        scanf("%d",&no);

        printf("Hangi dersi almak istiyorsunuz?\n";
        printf(" 1—Matematik 2—Tarih    3--İngilizce\n");
        printf("  ====>");
        scanf("%d",&sub);

        printf("Notu %d.\n", pnt[no-1][sub-1]);         //--2
}
```

- **Program açıklaması**

İki boyutlu bir dizi şekilde görüldüğü gibi bildirilmiş ve başlangıç değerleri atanmıştır. Ana hafıza içerisinde, dizi elemanları şekildeki gibi yerleştirilmişlerdir.

pnt[alan][alan], ilk alan öğrenci numarası için, ikinci alan ise ders adı için ayrılmıştır. Alanlar 0'dan başladığı için program içerisinde başlangıç değerine ulaşmak için girilen sayıdan 1 çıkartılır.



Şekil 1.11: Bellekteki iki boyutlu dizinin elemanları

	Matematik	Tarih	İngilizce
Öğrenci 1	**pnt[0][0]** 80	**pnt[0][1]** 90	**pnt[0][2]** 100
Öğrenci 2	**pnt[1][0]** 80	**pnt[1][1]** 75	**pnt[1][2]** 60
:	:	:	:
Öğrenci 30	**pnt[29][0]** 60	**pnt[29][1]** 100	**pnt[29][2]** 65

Şekil 1.12: İki boyutlu bir dizinin yapısı

UYGULAMA FAALİYETİ

Aşağıdaki sorulara ilişkin uygulama faaliyetini yapınız.

- Örnek 1.1 deki programı dizi sayısını sabit olmayacak şekilde yazınız. Sayı limitinin üst sınırı 49 olacaktır.
- Aşağıdaki gibi başlatılan bir dizinin yılın başlangıcından itibaren toplam gün sayısını gösteren programı yazınız. Kullanıcı ay ve gün değerlerini girecek ve yıl için döngü kullanabilirsiniz.
- int ay[12]={0,31,59,90,120,151,181,212,243,274,304,334};
- 1'den 50'ye kadar olan sayılardan rastgele olarak kabul eden ve bunları sıralama yöntemi kullanarak azalan şekilde sıralayan programı yazınız.
- Bir dizi içine 10 adet sayı yerleştirdikten sonra bu değerlerin aritmetik ortalamasını bulan programı yazınız.
- Başlatma yapılarak 10 elemanlı bir diziye isteğe bağlı 10 adet integer sayı giriniz. Bu dizi içerisindeki maksimum ve minimum sayıları bulan programı yazınız.

İşlem Basamakları	Öneriler
➢ Değişken tablosunu hazırlayınız. ➢ Akış diyagramını çiziniz. ➢ Programı yazınız. ➢ Yazdığınız programı derleyiniz. ➢ Programda hata var ise bunları gideriniz. ➢ Ekran görüntüsünü kontrol ediniz.	➢ Programda kullanacağınız değişkenlerin tipini belirleyiniz. ➢ Değişken isimlendirme kurallarına dikkat ediniz. ➢ Kullanacağınız verinin hangi dizi yapısına uygun olduğuna dikkat ediniz. ➢ Dizi sınırlarına dikkat ediniz. ➢ Akış diyagramı sembollerinden yararlanınız. ➢ Program satırlarının düzenli olmasına özen gösteriniz.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet,** kazanamadıklarınız için **Hayır** kutucuklarına (**X**) işareti koyarak kontrol ediniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Tek boyutlu integer dizi tanımladınız mı?		
2. Dizinin ilk değerini okurken döngünün başlangıç değerini 0 yaptınız mı?		
3. Scanf komutu ile dizi değişkenlerine döngü içinde değer aktardınız mı?		
4. Printf komutu ile dizi değerlerini ekrana yazdırdınız mı?		
5. Dizinin içindeki bir değeri program ile sildirdiniz mi?		
6. Dizilerin boyutlarını ihmal ederek tanıma yaptınız mı?		
7. Dizi içindeki sayısal değerleri büyükten küçüğe doğru sıralama algoritması oluşturdunuz mu?		
8. Bir dizi içersindeki en büyük ve en küçük değeri bulma akış diyagramı çizdiniz mi?		
9. İki boyutlu dizi tanımladınız mı?		
10. Teknolojik kurallara uygun bir çalışma gerçekleştirdiniz mi?		
11. Süreyi iyi kullandınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda **"Hayır"** şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız **"Evet"** ise "Ölçme ve Değerlendirme"ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.

1. ( ) **veri [10]** /* dizisi 10 elemanlıdır.*/

2. ( ) int **a[5]** /* a dizisi bellekte 10 byte yer kaplar.*/

3. ( ) no**[6]=14** /* no dizisinin 6. elemanı 14 sayısıdır.*/

4. ( ) int **rakam[100], a , b** /* tanımlaması yanlıştır.*/

5. ( ) veri**[2] = sayi[2];** /* sayi dizisinin 3. elemanına veri dizisinin 3. elemanı atanıyor.*/

6. ( ) for **(i = 0; i < 10; i++)** /* ifadesi 10 elemanlı sayı dizisinin tüm elemanlarına sıfır
sayi[i]= 0; /* değeri atar */

7. ( ) **i = 1;**
while (i <= 100)
{
scanf("%d", &fiyat[i]);
i++;
} /* fiyat dizisinin tüm elemanları için klavyeden değer alınır */

8. ( ) int **b[5] = {1, 2, 3, 4, 5, 6};** /* ifadesi derlemede hata mesajı verir. ***/**

9. ( ) **int numara [ ] = {1, 2, 3, 4, 5};** /* ifadesi kullanıldığında küme parantez içindeki */
/* değerleri otomatik olarak kabul eder. */

10. gecici**= a[i];**
a[i] = a[j];
gecici = a[j]; /* ifadesiyle a dizisinin i nolu değeri j nolu değeri ile yer
/* değişiyor */

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ- 2

AMAÇ

Bilgisayar programlamada karakter ve karakter katarlarını doğru bir şekilde kullanabileceksiniz.

ARAŞTIRMA

- Sevgili öğrenci, bu öğrenme faaliyetinden önce aşağıdaki hazırlıkları yapınız.
- C programlamada karakter katarları hakkında araştırma yapınız.

2. KARAKTER KATARLARI

Şimdiye kadar genel olarak sayısal veriler ile çalıştık. Karakter ve karakter katarları bilgisayarlarda farklı şekillerde ele alınırlar. Bu öğrenme faaliyetinde karakter tipindeki veriler anlatılacaktır.

2.1. Karakter ve Karakter Kodu

Örnek:

Klavyeden girilen bir karakteri ve kodunu görüntüleyen programı yazınız.

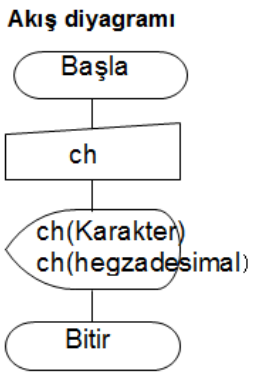
Ekran görüntüsü

```
Bir karakter giriniz ==>J
Karakter=J  Kod=4A
```

Değişken tablosu

Değişken	tür	Kullanım
ch	char	Karakterin Saklanması

Akış diyagramı



Şekil 2.1: Ekran görüntüsü, değişken tablosu ve akış diyagramı

```
/* Ornek 2.1 */

#include <stdio.h>

main()
{
        char ch;

        printf("Bir karakter giriniz ==>")
        scanf("%c",&ch);

        printf("Karakter=%c  Kodu=%X", ch,ch);
}
```

```
/* Ornek 2.1 */

#include <stdio.h>

main()
{
        char ch;

        printf("Bir karakter giriniz ==>")
        scanf("%c",&ch);

        printf("Karakter=%c   Kodu=%X", ch,ch);
}
```

- **Program açıklaması**

Şimdiye kadar öğrendiklerimiz gibi, printf ve scanf fonksiyonlarında tip dönüşüm belirteçlerini kullanacağız. Bir karakter, hafızaya karakter kodu formunda kaydedilir. Bir karakterin kodu %x tip dönüşüm belirteci ile görüntülenebilir.

(1) Karakter kod tablosu

Karakterleri belirtmek için kullanılacak veriler çok fazladır. Bunu anlayabilmek için, A karakterini alfabeyi bilmeyen bir yabancıya açıklamaya kalkmanın ne kadar zor olduğunu düşününüz. Bu yüzden bilgisayarlar her birisi kendi koduna sahip tüm karakterlerin şekillerini içerirler. Sadece 1 byte bir karakteri saklayabilmek için yeterlidir. 1 byte bize 2^8=256 adet farklı karakter üretebilir.

Karakterler için çeşitli kodlar vardır. Şekil 2.1 yaygın olarak kullanılan bir tablo olan ASCII (American Standard Code for Information Interchange) kod tablosunu göstermektedir. Tablo satır ve sütunlara sahiptir. Karakterlerin sayısal değerinin bulunması için önce sütun daha sonra satır bilgisi okunur. '**j**' karakterinin kodu şekilde görüldüğü gibi onaltılık sistemde olarak 4A değerine eşittir.

Sütun 0 ve 1 in kodu karakter kodları için değildir. Bunlar ekran kontrolü ve veri transfer kontrolü için kullanılır. Bütün bunlar kontrol kodu olarak ifade edilirler. ASCII tabloda 3. sütunda gösterilen sayısal değerler aynı zamanda karakter koduna sahiptirler. Örneğin '3' karakterinin kodu 33 sayısıdır.

sütun / satır	0	1	2	3	4	5	6	7
0	NUL	DLE	Space	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	TAB	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	?	?	O	_	o	

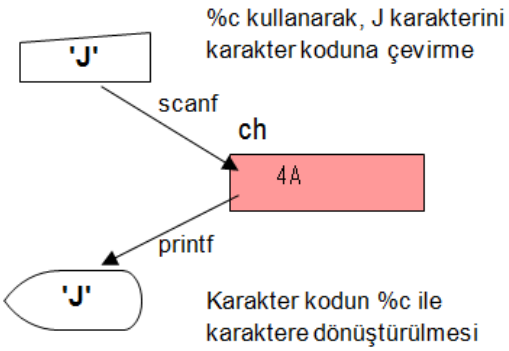
Gölgelendirilmiş bölümler daha sık kullanılır. Şekildeki örneğe göre 'J' nin karakter kodu $4A_{16}$ değeridir.

Şekil 2.2: ASCII karakter kodları

Karakterden koda dönüştürmede veya tersi işlemde scanf ve printf fonksiyonlarında kullanılan **%c** tip dönüşüm belirteçleri kullanılmıştır.

(2) Karakterlerin printf ve scanf fonksiyonunda kullanımları

```
char ch;
scanf("%c",&ch);      //--1
printf("%c",ch);      //--2
```



Şekil 2.3: Karakter alma ve gösterme

1.gösteriminde yazılan karakter **%c** kullanılarak karakter koda dönüştürülmüştür. Daha sonra karakter **printf** fonksiyonunda **%c** komutuyla gösterilmektedir.

%c yerine **%X** kullanarak karakter kodu hegzadesimal formda gösterebiliriz.

```
printf("%X",ch); veya %d ile desimal olarak
printf("%d",ch);
```

(3) Karakter sabiti

Bir karakteri tek tırnak arasına aldığımızda, bu karakter bir karakter sabiti olarak kabul edilirler. Karakter sabiti karakter koduna sahiptir.

Örneğin,

ch = 'D'; komutu ch değişkenine D nin kodunu aktarır.

Karakter katarları aşağıdaki şekilde başlatılabilir.

char char[ ]={'A','B','C'};

Tek tırnak işareti, karakterin, karakter kodundan önde geldiği durumlarda da kullanılır.

Bir problemde scanf fonksiyonunda %c kullanıldığında, bazen beklenilmeyen sonuçlar doğabilir.

Örneğin, Aşağıdaki gibi 2 tane scanf fonksiyonu yazdığımızda,

```
scanf("%d",&fiyat);
scanf("%c",&kayit);
```

LF kodu (Kontrol kodu- 0A ASCII), desimal sayı girildikten ve enter tuşuna basıldıktan sonra üretilen kod ch değişkenine atanır. Bu yüzden, sonuç olarak ikinci scanf fonksiyonu atlanır.

Bu durumu engellemek için ikinci scanf fonksiyonunda şöyle bir düzenleme yapmalıyız.

```
scanf("%c%c",&tutar,&kayit);
```

Bu durumda LF kodu dummy değişkenine atanır. Eğer karışık gelmez ise bu kodda kullanılabilir.

```
scanf("%c%c",&kayit,&kayit)
```

2.2. Karakter Katarları

Bir karakter dizisinde saklanan karakter katarına string adı verilir.

Örnek:

10 karaktere kadar bir string ifadeyi klavyeden okuyup her bir karakteri ve kodunu görüntüleyen programı yazınız.

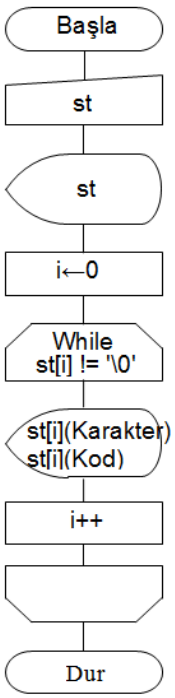
Ekran görüntüsü

```
String ifade giriniz ==>bitir
String=bitir
Karakter=s Kod=73
Karakter=t Kod=74
Karakter=o Kod=6F
Karakter=p Kod=70
```

Değişken Tablosu

Değişken	Tür	Kullanım
St[11]	int	String saklama
i	int	Kod değişkeni

Akış diyagramı



Program

```
/* Örnek 2.2 */

#include <stdio.h>

main()
{
    char st[11];
    int  i;

    printf"string ifade giriniz==>");
    scanf("%s",st);

    printf("String=%s\n",st);
    i=0;
    while(st[i] !='\0')
    {
        printf("Karakter=%c Kod=%X \n",st[i]);
       i++;
    }
}
```

- **Program açıklaması**

String ifade printf ve scanf fonksiyonlarında %s ile kullanılır. Bir karakter katarında tutulan string ifadenin elemanlarına ulaşabiliriz. Detaylı açıklamalar program yapısında anlatılacaktır.

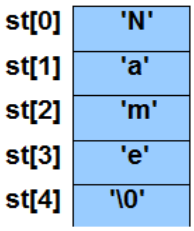
(1) Bir karakter dizisindeki string ifade

String ifadedeki her bir karakter, 1 karakterlik boşluğa gerek duyar. Bu yüzden string ifade için karakter katarını şu şekilde hazırlarız.

char st [7];

C dilinde, String ifadenin sonu 0 ile sonlandırılır. 0 değeri ASCII kod tablosunda BOŞ(NULL) değeridir. Bu karakter, NULL karakter olarak adlandırılır ve '\0' ile gösterilir. Hafızanın yapısının gerektirdiği '\0' değer için, karakter dizisinin boyutu (string ifadenin boyu + 1) şeklinde ayarlanmalıdır. Bir string ifade için kullanılan dizide, \0 değerinden sonraki değerlerin string ifade için bir önemi yoktur.

Aşağıda bu durum st[5] ve st[6] alanlarıdır.



Şekil 2.4: Karakter katarında string ifade

(2) Printf fonksiyonu ile string ifadenin yazdırılması

Bir karakter dizisindeki string ifadeyi şu şekilde gösterebiliriz.

```
printf("%s",st);
```

Tip dönüşüm belirteci olarak %s ve dizi ismi olarak st kullanılmıştır. Dizi ismi, parantez ve parametresiz kullanıldığında, dizinin en üst adresine sahip olur. Böylce %s tip dönüşüm belirteci ile printf fonksiyonu, birer birer en üstten başlamak üzere \0 kadar olan bütün değerleri yazar.

(3) Scanf fonksiyonu ile string ifadeler

Scanf fonksiyonundaki tip dönüşüm belirteci **printf** fonksiyonundaki ile aynıdır. Bu yüzden şu şekilde yazılabilir.

```
scanf("%s",st);
```

st dizi isminin kullanılmasından dolayı , & adres operatörüyle aşağıdaki şekilde yazamazsınız.

```
scanf("%s",&st);
```

Takip eden kod hatalı değildir fakat biraz hantal bir yapıya sahiptir.

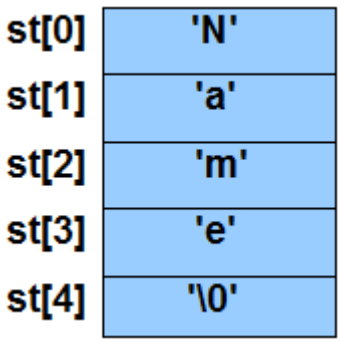
```
scanf("%s",&st[0]);
```

Scanf fonksiyonu, dizinin en üst noktasından itibaren değerleri okur ve sonuna \0 değerini ekler. Bir string ifadeyi içerisindeki boşluklarla girdiğinizde, scanf fonksiyonu sadece birinci boşluğa kadar okuyabilir.

(4) String sabiti ile başlatma

Eleman değerini yazmak için, karakter dizisinin başlatılması şu şekilde de gerçekleştirilebilir.

char st[ ]="Name";



Şekil 2.5: String ifadesiyle başlatma

String ifadede başlatma işlemini şu şekilde bir tanımlama ile yapamayız.

char st[5];
st="name";.

Bir string ifadenin başka bir diziye kopyalanması yada karakter dizisine string sabiti olarak ayarlanması için **strcpy** fonksiyonu kullanılır. Bu komutu daha sonra çalışacağız.

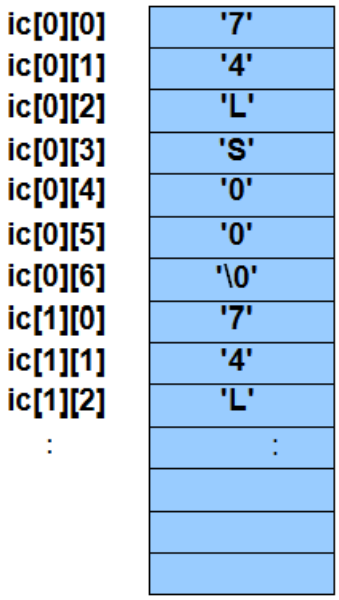
Karakter katarı str yi şu şekilde başlatabiliriz,

char str[]=” artık kendi programlarımı yazabiliyorum.”

(5) İki boyutlu diziler ve string ifadeler

İki boyutlu karakter dizisi bir String’ten daha fazlasını saklayabilir. İlk parametre, 0 dan itibaren string ifadenin sırasını gösterir. İkinci parametre 0 dan itibaren string ifadedeki karakterlerin sırasını göstermektedir. Her string ifade, sonunda **\0** değerine sahip olmalıdır.

Şekil 2.6 de, "74LS00", ve "74LS04" entegrelerin isimleri bellekte saklanmaktadır.



Şekil 2.6: Bir string ifadenin başlatılması

İki boyutlu karakter katarı aşağıdaki şekilde ifade edilebilir.

```
char dizi_isim[string sayısı][Stringlerin maksimum uzunluğu+1];
```

String ifadeye ulaşmak için, her string ifadenin en üst adresini kullanırız. ”74LS00” değerini şekildeki gibi göstermek için örneğin, printf fonksiyonu aşağıdaki gibi kullanılır.

printf("%s",&ic[0][0]), bir başka ifadeyle,
printf("%s",ic[0]); (& ve ikinci parametreyi kullanmadan)

Tek boyutlu bir dizideki dizi ismi olarak en üst adresi gösterebiliriz.

ic[0], ic[0][0]-ic[0][6]‘ nin en üst adresini ifade etmektedir. Benzer şekilde, scanf fonksiyonu aşağıdaki gibi yazılabilir.

scanf("%s",ic[0])

Aşağıdaki program, iki boyutlu bir karakter dizisinin nasıl başlatıldığını gösterir.

Program

```
/*  İki boyutlu bir dizi için program örneği*/

include <stdio.h>
main()
{
    int i;
    char ic[3][7]={"74LS00","74LS02","74LS04"};

    printf("Hangi entegreyi sipariş etmek istiyorsunuz? \n");
    printf(" 1.%10s 2.%10s 3.%10s\n",ic[0],ic[1],ic[2]);
    printf (" numarasını giriniz (1-3)  ==>");
    scanf("%d",&i);

    printf ("sipariş etmiş olduğunuz entegre.. %s. \n", ic[i-1]);
}
```

Değişkenleri tabloda görüldüğü gibi tanımlayabiliriz. Bundan sonra akış diyagramlarında değişken tanımlamak yerine, bunun gibi değişken tabloları kullanacağız.

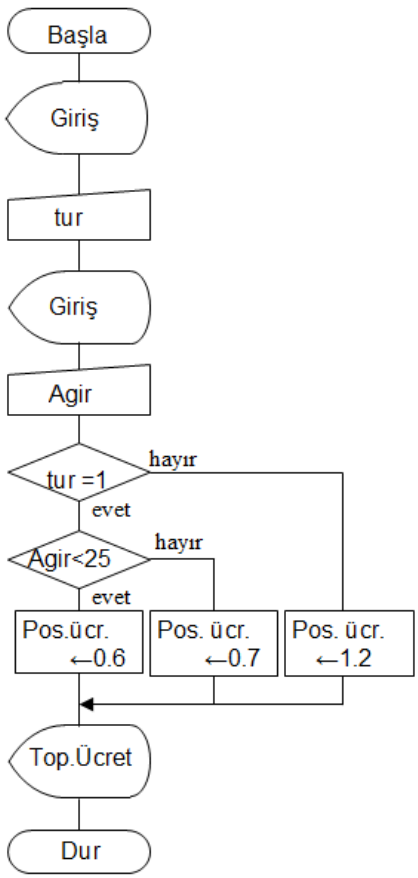
2.3. Çok Yönlü Karar Yapısı

İki veya daha fazla durum için çok yönlü karar yapıları kullanılır. Bu yapıda, bir ifade diğer ifadenin durumuna bağlıdır.

Ekran görüntüsü

```
Tür (1.Düzenli 2.Düzensiz)==>1
Ağırlık(g)                ==>20
Posta ucreti toplam  0.6 YTL
```

Akış diyagramı



Program

```
/* Örnek 2.3 */

#include <stdio.h>

main()
{
   int tur,agir;
   float posta;
   printf("Tür(1.Duzenli 2.Duzensiz)==>");
   scanf("%d",&tur);
   printf("Ağırlık(g)==>");
   scanf("%d",&agir);
   if ( tur ==1){
        if (agir <= 25)
          posta=0.6;
        else
        posta=0.7;
   }
   else
        posta=1.2;

printf("Posta ucreti toplam %3.1f YTL\n", posta);
}
```

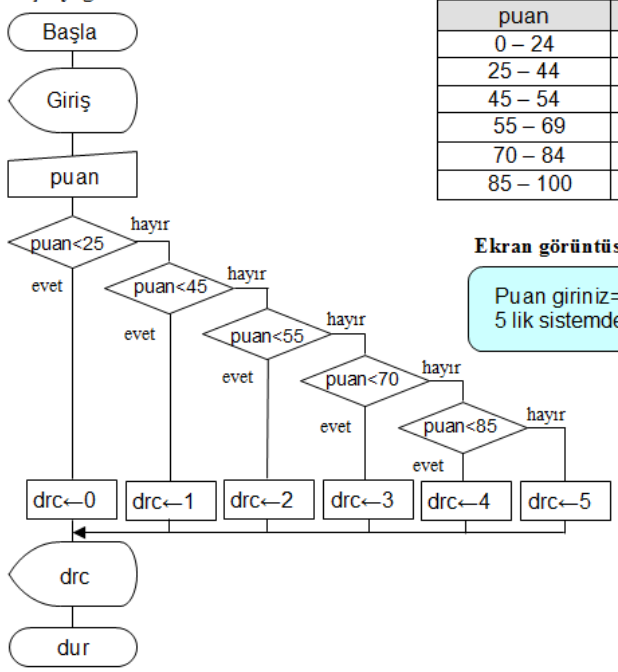
- **Program Açıklaması**

Posta türünün ve ağırlığının scanf fonksiyonuyla klavyeden girilmesi sağlandıktan sonra program öncelikle girilen ağırlığın düzenli mi düzensiz mi olduğunu kontrol eder. Düzenli olduğunda, tekrar bir if bloğu ile 25 gramın altında olup olmadığı kontrol edilir. Eğer bir if bloğunda başka bir if bloğu var ise buna iç içe geçmiş if bloğu adı verilir.

Örnek:

Aşağıdaki tablodaki puan aralıklarına göre klavyeden girilen notun 5'lik sistemdeki karşılığını veren program yazınız.

Akış diyagramı



puan	derece
0 – 24	0
25 – 44	1
45 – 54	2
55 – 69	3
70 – 84	4
85 – 100	5

Ekran görüntüsü

```
Puan giriniz==>48
5 lik sistemdeki karşılığı 2
```

Program

```
/* örnek 2.4 */
#include<stdio.h>

main()
{
    int puan,drc;
    printf("Puan giriniz==>");
    scanf("%d",&puan);

    if (puan<25)                          //--1
       drc = 0;
       else if (puan<45)                  //--2
          drc = 1;
          else if (puan<55)               //--3
             drc = 2;
             else if (puan<70)            //--4
             drc = 3;
                 else if (puan<85)        //--5
                     drc = 4;
                 else                     //--6
                 drc = 5;
printf("5'lik sistemdeki karsiligi %d.\n",drc);
}
```

- **Program Açıklaması**

Puanın girilmesinden sonra bu program, puan <25 şartının sağlayıp sağlamadığını kontrol eder. Bu karşılaştırma, doğru sonucu veriyorsa program bunu birinci değişken olarak atar; yanlış ise puan, ikinci if koşulu ile tekrar kontrol edilir. Bu durum son if koşuluna kadar devam eder. Son durum da yanlış sonuç verirse son if koşulundan sonraki else icra edilir. Bu else veya else bloğu, gerekli değilse ihmale edilebilir. İç içe if yapısı ile çoklu seçim yönteminin kullandığı yapı konusunda dikkatli olunması gerekir. Çoklu yönlü karar yönteminin kod yazma aşamasında karar ifadelerinin aynı hizada değil de belirli bir boşluk kadar içeriden başlaması, programı daha anlaşılır hâle getirir.

2.4. Çoklu Karar Yapısı

Çok yönlü karar yapısındaki durumların kontrolü birbirine bağlı şartlara bağlıdır. Çoklu karar yapısında (switch-case) şartların hiçbir önceliği yoktur.

Örnek:

Aşağıda yer aldığı şekilde notları derecelendiren programı yazınız. Gerekli olan tablo aşağıda verilmiştir.

puan	derece
8-10	1
7	2
6	3
0 – 5	4

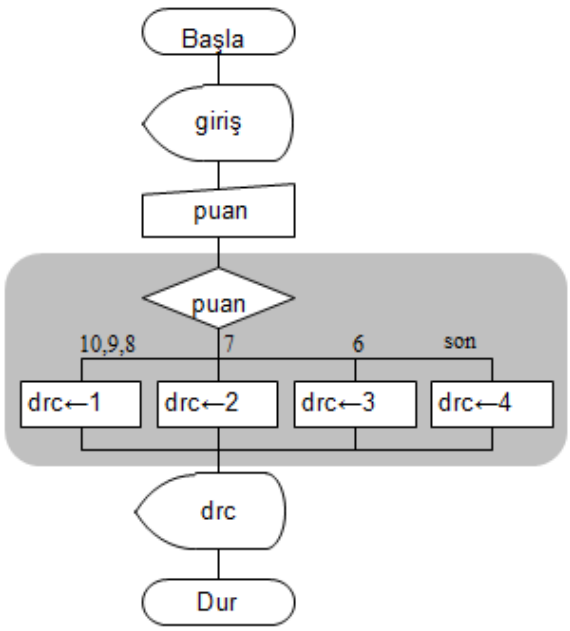
Ekran görüntüsü

```
Puan giriniz==>7
Dereceniz  2.
```

Değişken Tablosu

Değişken	Tür	Kullanım
puan	int	Not
drc	int	Nota karşılık gelen derece

Akış diyagramı



Program

```
/* Örnek 2.5*/

#include<stdio.h>
main()
{
     int not1,drc;
    printf("Puan giriniz==>");
    scanf("%d",&puan);
    switch (puan) {
        case 10:
        case 9:
        case 8:
           drc=1;
           break;
        case 7:
           drc=2;
           break;
        case 6:
           drc=3;
           break;
        default:
           drc=4;
    }
    printf("Dereceniz  %d. \n",drc);
}
```

➢ **Program açıklaması**

Puan değişkeninin değerine bağlı olarak yapılan çok yönlü karar yapısı akış diyagramında gösterilmiştir. "puan" değişkeninin değeri 10,9 veya 8 olduğunda drc←1 işlemi gerçekleştirilecektir ve 7 değeri için drc ←2 değeri aktarılacaktır. Benzer şekilde, 6 için drc←2 olacaktır. Son olarak puan değeri, son değerini**(default)** alacak ve bu şekilde işletilecektir.

Tüm bunlar program örneğinde C dilindeki switch-case yapısı içinde kodlanmıştır. Bu yapıda, "puan" değişkeni "switch" ifadesinden sonra parantez içerisine alınır ve "case" komutundan sonra kontrol edilecek değişkenin değerleri verilir. Bu değerlerden sonra "**:**" karakteri kullanılmak zorundadır.

Break komutu, komutların çalışmasını durdurarak "switch-case" yapısından dışarı çıkılmasına imkân verir. Caseden sonra kullanılan tüm komutlar, break komutu ile karşılaşıncaya kadar işletilir. Bu programda, **case 10: case 9:** ve **case 8**: ifadelerinden sonra break komutu yoktur. Tüm bu değerler, aynı komutun icra edilmesini sağlar. (drc=1;)

Switch-case yapısında, case ifadesinden sonra integer bir değer gibi 'A' karakterini de yazabiliriz. A karakteri, tek tırnak arasında olmalıdır. Bu tür bir yazım, karar yapılarında karakter kodunun kontrol edilmesini sağlar. Karakter kodu hakkında ileride daha detaylı durulacaktır.

```
char  ch1;
ch1='A';
```

Bir karakteri saklayan değişken, bir karakter tip değişkenidir ve char tipi ile tanımlanır. Scanf ve printf komutundaki ve tip dönüşüm belirteci karakter için %c'dir.

Aşağıdaki program, bir malın kodunu kontrol ederek malın cinsini yazar.

```
#include <stdio.h>
main()
{
    char kod;
    printf("Kodu giriniz==>");
    scanf("%c",&kod);

    switch(kod){
    case 'A':
          printf("Kod %c --- TV\n",kod);
    break;
    case 'B':
        printf("Kod %c --- Radyo\n",kod);
    break;
default:
    printf("Kod %c --- diğer\n",kod);
   }
}
```

Ekran görüntüsü

```
Kodu giriniz==>B
Kod  B --- Radyo
```

UYGULAMA FAALİYETİ

Aşağıdaki sorulara ilişkin uygulama faaliyetini yapınız.

- Klavyeden girilen iki tam sayıdan büyük olanı ekrana yazan programı yazınız.

- İki sınavın notlarının klavyeden girilmesi istenmektedir.Öğrencinin başarılı olabilmesi için gereken şartlar şunlardır:

  - Toplam sonuç 140 ve daha büyük
  - İki not da 65'ten küçük olmayacak.

Bu şartlara göre ekrana başarılı veya başarısız yazan programı yazınız.

- Aşağıdaki tablo, bir otobüs hattındaki uzaklıklara göre yol ücretini göstermektedir. Bu tabloya göre klavyeden uzaklığı km cinsinden girilen bir yere gitmek için ödeyeceğimiz yol parasını hesaplayan programı yazınız.

Uzaklık	Yol ücreti
0.0 - 9.9km	**2 YTL**
10.0 - 29.9km	**3 YTL**
30.0 - 49.9km	**4 YTL**
50km -	**5 YTL**

- Girilen gelirin büyüklüğüne göre aşağıda belirtilen oranlarda gelir vergisi alınacaktır. Klavyeden girilen gelire göre ödenmesi gereken gelir vergisini hesaplayan programı yazınız.

Gelir vergisi için tanımlanan kurallar aşağıda verilmiştir.

Gelir 10,000 YTL den düşük ise vergi yok.
10,000 YTL- 299,999 YTL 5%
300,000 YTL- 499,999 YTL 8%

Bundan sonra her 200,000 YTL artışı vergi oranını 1% artıracaktır.

Örneğin 800,000 YTL için 10% olacaktır.

Not: Gelir vergisi maksimum 80 % olabilir.

Aşağıdaki işlem basamaklarını takip ederek belirtilen sorulara ilişkin uygulama faaliyetini yapınız.

İşlem Basamakları	Öneriler
➢ Değişken tablosunu hazırlayınız. ➢ Akış diyagramını çiziniz. ➢ Programı yazınız. ➢ Yazdığınız programı derleyiniz. ➢ Programda hata var ise bunları gideriniz. ➢ Ekran görüntüsünü kontrol ediniz.	➢ Programda kullanacağınız değişkenlerin tipini belirleyiniz. ➢ Değişken isimlendirme kurallarına dikkat ediniz. ➢ Akış diyagramı sembollerinden yararlanınız. ➢ Program satırlarının düzenli olmasına özen gösteriniz. ➢ Matematiksel işlem önceliklerine dikkat ediniz. ➢ Karar ifadelerinin belirlenen şartlara uygun olmasına dikkat ediniz.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet,** kazanamadıklarınız için **Hayır** kutucuklarına (**X**) işareti koyarak kontrol ediniz.

Performans Değerlendirme	Evet	Hayır
1. Karakter katarı tanımladınız mı?		
2. Scanf komutu ile karakter katarına değer atatınız mı?		
3. Printf komutu ile karakter katarının harf harf yazdırdınız mı?		
4. Printf komutu ile karakter katarını tamamını yazdırdınız mı?		
5. Karakter katarının bir harfini %X ve %d ile ekrana yazdırdınız mı?		
6. İki boyutlu karakter tanımlaması yaptınız mı?		
7. İki boyutlu karakter katarını ekrana yazdırdınız mı?		
8. If-else komutları ile çok yönlü karar yapısı oluşturdunuz mu?		
9. Switch case komut yapısı ile çoklu karar yapısı oluşturdunuz mu?		
10. Break komutu ile çoklu karar yapısında çıkış yaptınız mı?		
11. Teknolojik kurallara uygun bir çalışma gerçekleştirdiniz mi?		
12. Süreyi iyi kullandınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda **"Hayır"** şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız **"Evet"** ise "Ölçme ve Değerlendirme"ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. a değişkeninin değeri b değişkeninin değerine eşit olduğunda hangi ifade kullanılır?
A) if(a>15)
B) if(a<15)
C) if(a=b)
D) if(a= =b)

2. a değişkeninin değeri 15 veya yukarısı olduğunda
A) if(a>=15)
B) if(a!=15)
C) if(a=!15)
D) if(a=>15)

3. a değişkenin değeri 15'ten küçük olduğunda
A) if(a>15)
B) if(a<15)
C) if(a=b)
D) if(a= =b)

4. a değişkeninin değeri 15 değerinden büyük olduğunda
A) if(a>15)
B) if(a<15)
C) if(a=b)
D) if(a= =b)

5. a değişkeninin değeri 15'ten farklı olduğunda
A) if(a>=15)
B) if(a!=15)
C) if(a=!15)
D) if(a=>15)

6. Sayı değişkeninin 10'a eşit olduğu ve toplam değişkeninin 20'den küçük olduğu durumlarda **"yanlıs"** kelimesini basan ifadeyi tamamlayınız.

if ((sayi ___10) _____ (toplam ___20))
printf("yanlıs.");

7. not1 değişkeninin 80'den fazla veya not2 değişkeninin 60'tan fazla olduğu durumlarda **"dogru"** kelimesini basan ifadeyi tamamlayınız.

```
if ((not1 ___80) ____ (toplam ___60))
printf("dogru ");
```

8. not1 değişkeninin 20'den küçük olmadığı ve 40'tan büyük olmadığı durumda **"geçerli sayı "** kelimelerini basan ifadeyi tamamlayınız.

```
if ((not1 ___20) ____(not1 ___40))
printf("gecerli sayi");
```

9. Aşağıdaki ifadeyi switch komutunu kullanarak tamamlayınız.

```
if( urun == 'A' )
    fiyat = 200;
else if ( urun == 'B' )
    fiyat = 50;
else if( urun == 'C' )
    fiyat = 600;
else
printf("Bilinmeyen urun -->%c\n", urun );
```

```
switch( _________ )
{
   case ____:  fiyat =200; break;
   case 'B'    : fiyat= 50; break;
   case 'C'    : fiyat =600; break;
   ________ printf( "Bilinmeyen urun -->%c\n", urun ); break;
}
```

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ- 3

AMAÇ

Bilgisayar programlamada döngü yapılarını doğru bir şekilde kullanabileceksiniz.

ARAŞTIRMA

- Bu öğrenme faaliyetinden önce aşağıdaki hazırlıkları yapınız.
- C programlama dilinde döngü yapıları hakkında araştırma yapınız.

3. DÖNGÜLER

Döngü (loop), programlamada en önemli yapılardan birisidir. Bazen birçok verinin çıktısının alınmasında veya kalem kâğıtla yapamayacağımız birçok işlemin yapılmasını sağlayan araçtır. Döngü yapısını tasarlamak, program yaparken karşılaşacağımız en zor ve en önemli durumlardan birisidir. Bu yüzden çok miktarda pratik yapmaya ihtiyacınız vardır.

3.1. While Döngüsü

Örnek:

"Merhaba!" kelimesini ekrana 5 kez yazan programı yapınız.

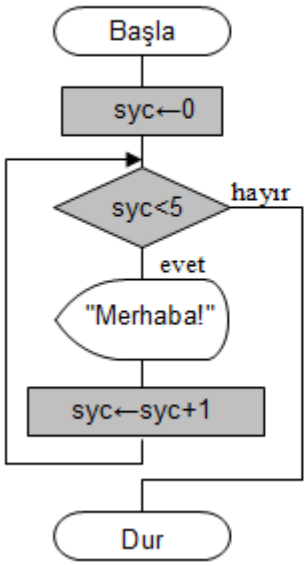
Değişken tablosu

Değişken	Tür	Kullanım
syc	int	Sayaç

Ekran görüntüsü

```
Merhaba!
Merhaba!
Merhaba!
Merhaba!
Merhaba!
```

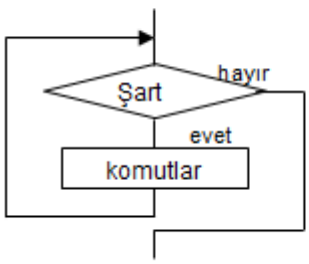
Akış diyagramı **Program**




```
/* Örnek 3.1 */

#include <stdio.h>

main()
{
    int syc;

    syc=0;
    while(syc < 5)
    {
        printf("Merhaba!\n");
        syc = syc + 1;
    }
}
```

➢ **Program açıklaması**

Genel olarak döngü oluşturmak için kullanılan komutlardan birisi while yapısıdır. Akış diyagramında görüldüğü gibi şart ifadesi döngünün içerisine girmeden önce kontrol edilmektedir. Şart ifadesi doğru olduğunda, döngü içerisindeki komutlar şart ifadesi yanlış oluncaya kadar icra edilecekleridir. Diğer durumda içerdeki komutlar işletilmeyecektir. Bu duruma işletilen durum adı verilir. Bu komutun yapısı, if yapısı ile aynı şekildedir. Aşağıdaki tabloya göre akış diyagramındaki while komut yapısı kolaylıkla koda dönüştürülebilir.

Akış Diyagramı	C dilindeki karşılığı
	**while(Şart ifadesi)** **{** **Komutlar** **}**

Şekil 3.1: While yapısı

İçinde döngü kullanılması gereken bir program tasarlanırken ilk olarak döngünün hangi değerden başlayacağını, döngünün nerede sonlandırılacağını ve döngünün nasıl tekrar edileceğini düşünmeliyiz. Programı yukarıdan aşağıya doğru satır satır yazmaya

çalıştığınızda istediğiniz sonuca elde etmeniz mümkün değildir. Program yapmak bulmaca çözmek gibidir. Bu yüzden öncelikle koyması veya düşünülmesi kolay olan parçaları tamamlamamız gerekir. Bu şekilde bir çalışmayı alışkanlık haline getirdiğinizde yaptığınız programlarda mantıksal olarak hata yapmazsınız. Programımızda önemli değişkenleri tanımladıktan sonra döngü yapısını oluşturmalısınız.

Örnek programda, döngü yapısını incelemek istersek syc değişkeninin üç kere kullanıldığını görürüz.

1.	syc=0	döngüden önce
2.	while(syc < 5)	while komutunda
3.	syc = syc + 1	döngünün son satırında

syc = 0 değeri döngüden önce sadece bir kez işletilir. Bu durumda değişkene 0 değeri (başlangıç değeri) atanmıştır.

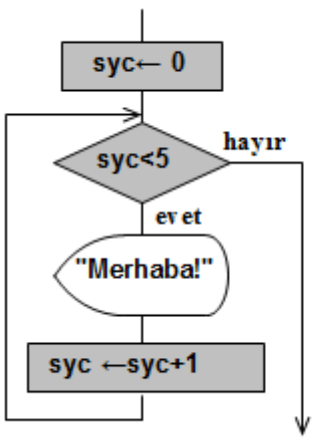
while(syc < 5) şartın kontrol edildiği durum

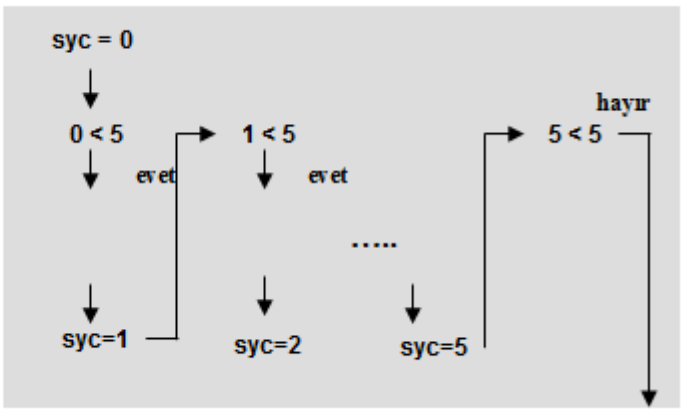
syc = syc + 1 matematikte imkânsız olan bu ifade, bilgisayar tarafından syc değerine sürekli 1 sayısının eklenmesi anlamındadır.

Döngü yapısı için bu üç ifade çok önemlidir.

- **Programın izlenmesi**

Aşağıdaki akış diyagramını incelerseniz tekrarın 5 kez yapıldığını görebilirsiniz.





Yukarıda adım adım yapılan yönteme izleme metodu (tracing) adı verilir. Adım adım izlemede kendinizi bilgisayarın yerine koyabilirsiniz. Program yazımından önce akış diyagramı çizerek bu yöntemi kullanmak, döngü yapısını öğrenmeniz açısından önemlidir.

Programlarda sayaç kullanımı çok sık kullanılır. C dilindeki basit kullanımı aşağıdaki görülmektedir.

```
syc++
```

Bu ifade artırma operatörü kullanılmıştır. Aynı şekilde azaltma operatörü de vardır.

Operatör	Kullanım	Anlam
++ (Arttırma)	i++	i = i + 1
-- (Azaltma)	i --	i = i - 1

Şekil 3.2: Artırma ve azaltma operatörü

Örnek:

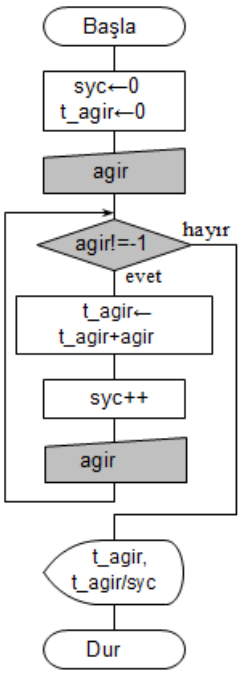
Girilen parça ağırlıklarına göre toplama ağırlık ve ortalama ağırlık hesaplamalarını yapan programı yazınız. -1 değeri girildiğinde bilgi girişi duracaktır.

Değişken	Tür	Kullanım
agir	float	Parça Ağırlığı
t_agir	float	Parçaların toplam ağırlığı
syc	int	Sayaç

Ekran görüntüsü

```
Parça ağırlığı(bitir--1)=>3.2
Parça ağırlığı(bitir--1)=>3.3
Parça ağırlığı(bitir-- 1)=>3.4
```

Akış diyagramı

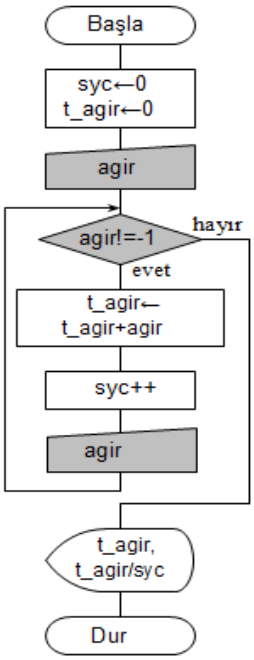


Program

```
/* Örnek 3.2 */

#include <stdio.h>

main()
{
      int syc=0;
      float agir, t_agir=0;
      printf("parca agirligi (bitir----1)=>");
      scanf("%f",&agir);
      while(agir !=-1)
      {
          t_agir = t_agir + agir;
          syc++;
          printf("parça agirliği (bitir----1)=>");
          scanf("%f",&agir);
       }
      printf("Toplam agirlik  =%6.1f\n",t_agir);
      printf("Ortalama agirlik =%6.1f",t_agir/syc);
}
```

Değişken	Tür	Kullanım
agir	float	Parça Ağırlığı
t_agir	float	Parçaların toplam ağırlığı
syc	int	Sayaç

Akış diyagramı



Program

```
/* Örnek 3.2 */

#include <stdio.h>

main()
{
      int syc=0;
      float agir, t_agir=0;
      printf("parca agirligi (bitir----1)=>");
      scanf("%f",&agir);
      while(agir !=-1)
      {
          t_agir = t_agir + agir;
          syc++;
          printf("parça agirliği (bitir----1)=>");
          scanf("%f",&agir);
       }
      printf("Toplam agirlik  =%6.1f\n",t_agir);
      printf("Ortalama agirlik =%6.1f",t_agir/syc);
}
```

1) Döngü Yapısı

Bu döngüyü tasarlarken anahtar değişken, ağırlık değerini kullanacak ağır değişkenidir. Bu durumda while döngüsünün şart ifadesi aşağıdaki şekilde yazılabilir.

agir != -1.

Akış diyagramında ağırlık girişi iki defa görülmektedir. Bir önceki programda olduğu gibi başlangıç değeri ve diğeri de artışın yapıldığı durumdur. Böylece bir önceki örnek ve bu örneği aynı yapıya sahip diye tanımlayabiliriz. Bu durumda iki defa değer girişi çok hoş durmamaktadır. Bu yapı bize programın kolayca yapılmasını sağlayacaktır.

2) Toplam Ağırlığın Bulunması

Toplam ağırlığın bulunması için iki hesaplamanın yapılması zorunludur.

t_agir = 0 (başlangıç değeri)

t_agir = t_agir + agir

İkinci hesaplama için yeni ağırlık değeri toplam ağırlığa eklenecek bir sonraki değeri ise yine bu toplam değere ilave edilerek sürekli bir toplama işlemi yapılacaktır.

3) Sayaç

Sayaç işlemi bir önceki programla aynı işlemi yapmaktadır.Bu durumda akış diyagramını tekrar incelemeniz ve döngünün girilen ağırlık sayısını sayıp saymadığını kontrol etmeniz gerekir.

4) Yazma

Toplam ağırlık ve sayaç kullanarak bizden istenilen toplam ağırlık ve ortalama ağırlık değerleri yazdırılabilir.

Örnek:

Aşağıdaki işlemin sonucunu bulan programı yazınız.

1+2+3……+50

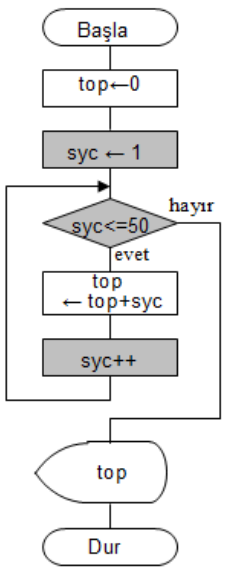
Değişken tablosu

Değişken	Tür	Kullanım
syc	int	Eklenen sayı
top	int	Toplam sonuç

Ekran görüntüsü

```
1+2+...+50=1275
```

Akış diyagramı



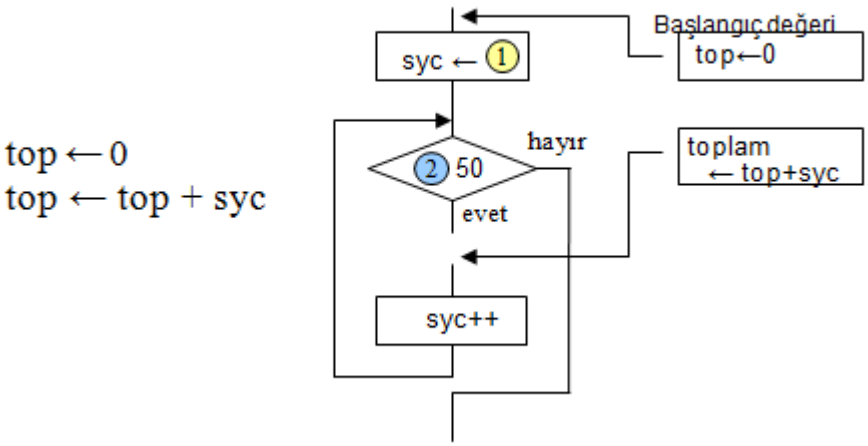
Program

```
/* Örnek 3.3 */

#include <stdio.h>

main()
{
    int top=0,syc;

    syc=1;
    while (syc<=50)
    {
        top=top+syc;
        syc++;
    }
    printf("1+2+...+50=%d\n", top);
}
```

➢ **Program açıklaması**

Bir döngü için kullanılan değişkene kontrol değişkeni adı verilir. Bu programda syc değişkeni, sayıları üretmek ve bunları kontrol değişkenine eklemek için kullanılmaktadır. Tasarım için öncelikle döngü işlemi şekil 3.3'de görüldüğü gibi hazırlanır. Tabloda 1 ve 2 bölümleri boş bırakılmıştır.

Daha önce de çalıştığımız gibi toplamın bulunması için iki elemana ihtiyacımız vardır.



Şekil 3.3: Artırma ve azaltma operatörü

Bu değişkenlerden birincisinin döngüden önce, diğerinin ise döngünün içerisinde olması düşünülmüştür. Son olarak 1 ve 2 numaralı bölümler son syc değeri için düşünülmelidir. 1'den 50'ye kadar olan sayıların toplamı bulunurken (50 dâhil) dikkat edilmesi gereken başlangıç değeri ve bitiş değerleridir. Başlangıç değeri olabilir ve 2 numaralı bölüm <= ile doldurulmalıdır. Son aşama için akış diyagramını dikkatlice takip etmelisiniz.

3.2. For Döngüsü

Örnek:

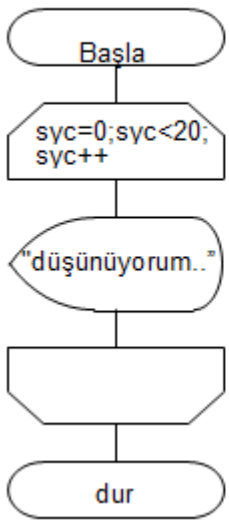
Düşünüyorum kelimesini alt alta ekrana basan programı yazınız.

Değişken tablosu		
Değişken	**Tür**	**Kullanım**
syc	int	Sayma işlemi(kontrol değişkeni)

Ekran görüntüsü

```
düşünüyorum...
düşünüyorum...
düşünüyorum...
düşünüyorum...
düşünüyorum...
```

Akış diyagramı

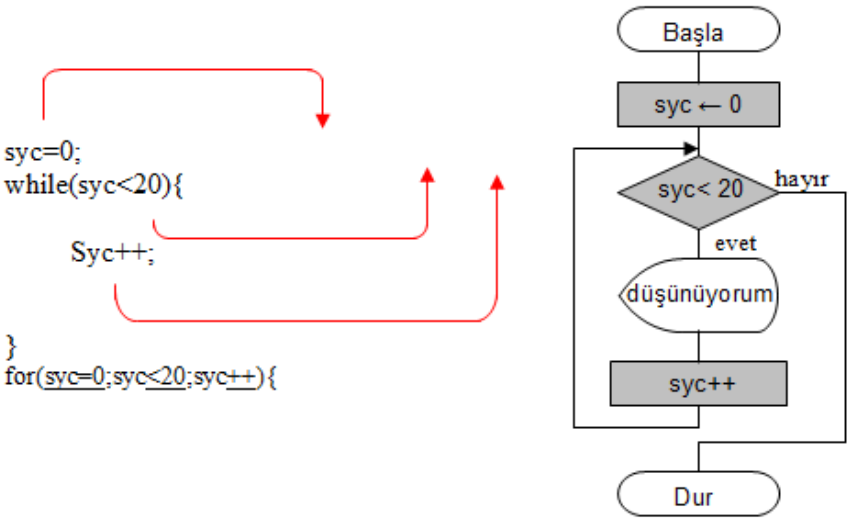


Program

```
/* Örnek 3.4 */

#include<stdio.h>

main()
{
    int syc;

    for(syc = 0; syc < 20;syc++)
    {
        printf("dusunuyorum...\n");
    }
}
```

- **Program Açıklaması**

Bu örnek while döngüsünün yerine for döngüsü ile yapılan temel sayma programıdır. Bunun için daha önceden tanıştığımız gerekli olan üç eleman vardır. Bunlar while yapısını kullanan aşağıdaki örnekle ilişkilidir.

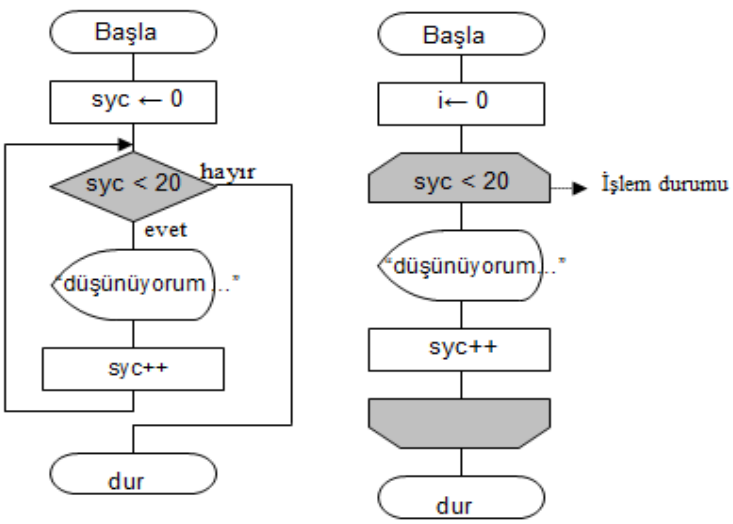


Şekil 3.4: While ve for döngü yapıları

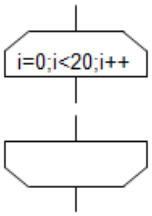
Aşağıdaki örnekte görüldüğü gibi for yapısı içinde sadece bir komut varsa süslü parantezler kullanmayabiliriz.

```
for(i = 0; i < 20;i++)
printf("düşünüyorum ...\n ");
```

Şimdiye kadar kullandığımız döngü akış diyagramlarını çizmek zaman almaktadır ve çizimi takip açısından biraz zordur. Bu yüzden bundan sonra sağdaki akış diyagramını döngüleri tanımlamak için kullanacağız.



Şekil 3.5: Döngü için akış diyagramları



For döngüsü için akış diyagramı

Bu tür bir kullanım döngü yapısını açıkça gösterdiği ve bizi önemli elemanları yazmayı unutmamızı engellediği için tercih edilen bir yöntemdir. Akış diyagramında, **for** döngüsü için gerekli şartlar sağda gösterildiği gibi yerleştirilmiştir.

Örnek:

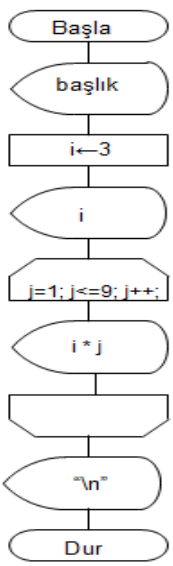
For döngüsü ile yandaki şekildeki gibi 3 değeri için çarpım tablosu oluşturan programı yazınız.

Ekran görüntüsü

```
     1  2  3  4  5  6  7  8  9
3    3  6  9  12 15 18 21 24 27
```

Değişken tablosu		
Değişken	**Tür**	**Kullanım**
i	int	3 değeri için
j	int	1 – 9 sayılarını üretmek için (Kontrol Değişkeni)

Akış diyagramı



Program

```
/* Örnek 3.5 */

#include<stdio.h>

main()
{
    int i,j;
    printf("   1  2  3  4  5  6  7  8  9\n");        //--1
    i=3;                                              //--2
    printf("%d",i);                                   //--3
    for(j  = 1; j<= 9;j++)                            //--4
        printf("%3d",i*j);
    printf("\n");                                     //--5
}
```

Program Açıklaması

1. İlk satırın ekranda gösterilmesi.
2. i değişkeni için 3 değerinin atanması.
3. \n elemanını kullanmadan i değişkeninin görüntülenmesi.
4. Değişen j değerine göre for döngüsü içinde (i × j) değerinin hesaplatılması
5. "\n" ile satırın değiştirilmesi.

3.2.1. İç İçe Döngüler

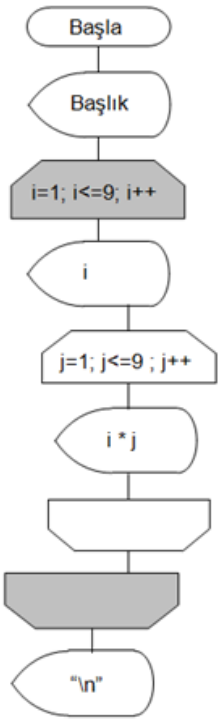
Örnek: Ekran çıktısında görülen çarpım tablosunun programını yazınız.

Ekran görüntüsü

```
    1  2  3  4  5  6  7  8  9
1   1  2  3  4  5  6  7  8  9
2   2  4  6  8  10 12 14 16 18
3   3  6  9  12 15 18 21 24 27
    :
9   9  18 27 36 45 54 63 72 81
```

Değişken tablosu		
Değişken	**Tür**	**Kullanım**
İ	int	1 – 9 arasındaki sayılar için (Dış döngünün kontrol değişkeni)
j	int	1 – 9 arasındaki sayıları için (İç döngünün kontrol değişkeni)

Akış diyagramı



Program

```
/* Örnek 3.6 */
#include <stdio.h>
main()
{
        int i,j;

        printf("  1  2  3  4  5  6  7  8  9\n");

        for(i = 1; i <= 9;i++)
        {
                printf("%d",i);
                for(j = 1; j <= 9; j++)
                        printf("%3d", i*j);
                printf("\n");

        }
}
```

Program Açıklaması

1 ile 9 arasındaki sayıları üretip kolon olarak kullanmak için dış döngüyü kullanabiliriz.

Örnek:

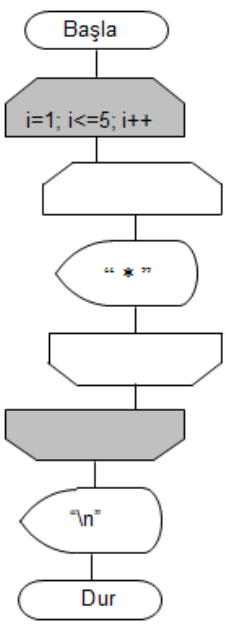
Alttaki şekilde görüldüğü gibi " * " karakterini ekrana basan programı yazınız.

Ekran görüntüsü

```
*
**
***
****
*****
```

Değişken tablosu		
Değişken	**Tür**	**Kullanım**
İ	int	Satır için kullanılan değişken
j	int	Sütun için kullanılan değişken

Akış diyagramı



Program

```
/* Örnek 3.7 */

#include <stdio.h>

main()
{
    int i,j;

    for(i = 1; i <= 5;i++)
    {
        for (j = 1; j <=i; j++)
          printf("*");
        printf("\n");
    }
}
```

Program Açıklaması

Şekildeki yapıyı "*" karakteri ile oluşturmak için iki döngüye ihtiyaç vardır. Programda i değişkeni satırlar için j değişkeni ise sütunlar için kullanılır. Dış döngüdeki i değişkeni 1'den 5 değerine kadar artan şekildedir. Döngüde i=1 olduğunda iç döngü j=1 değerinden başlayacak ve j=i➔1 olduğu için sadece 1 adet "*" karakteri yazacaktır.

Program 2. satıra printf("\n") komutu ile geçer. Bu durumda dıştaki döngüde i=2 olmuştur. İç döngüde ise j=i➔2 olduğu için 2 defa "*" karakterine basar.

Bu işlem dış döngüde i=4 değerine kadar tekrarlanır.

Örnek:

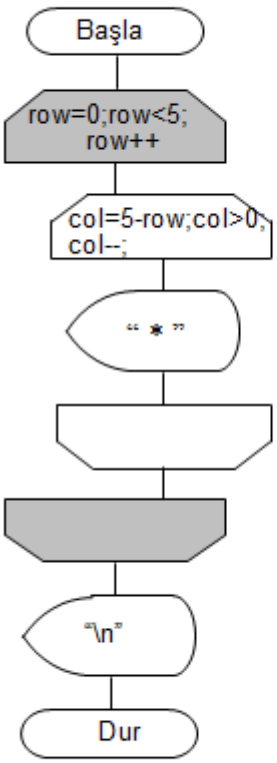
Aşağıdaki şekilde görüldüğü gibi " * " karakterini ekrana basan programı yazınız.

Ekran görüntüsü

```
*****
****
***
**
*
```

Değişken tablosu		
Değişken	**Tür**	**Kullanım**
row	int	Satır için kullanılan değişken
col	int	Sütun için kullanılan değişken

Akış diyagramı



Program

```
/* Örnek 3.8 */

#include<stdio.h>

main()
{
        int row,col;

        for(row=0;row<5;row++)
        {
                for(col=5-row;col>0;col--)
                        printf("*");
        printf("\n");
         }
}
```

Program Açıklaması

Bu örnek yapı olarak önceki örneğin benzeriydi. Program dıştaki döngüden başlar.Dış döngüdeki **row** değişkeni **0** değerinden başladığı için içteki döngüde **col** değişkeni **col=5-0** ➔5'den 1'e kadar 5 defa * karakterini yazar. Program 2. satıra **printf("\n")** komutu ile geçer. Bu durumda dıştaki döngüde **row=1** olmuştur. İç döngüde ise **col**=5-1➔4 değerinden 1 değerine kadar 4 defa **"*"** karakterine basar.

Bu işlem dış döngüde **row=4** değerine kadar tekrarlanır.

UYGULAMA FAALİYETİ

Aşağıdaki sorulara ilişkin uygulama faaliyetini yapınız.

- Birim fiyatları ve satış rakamları girilen malların toplam tutarını hesaplayan programı yazınız(klavyeden girilen verilerin sonlandırma işlemi 0 ile yapılabilir).

Birim fiyat	2000
Satış adedi	12
Birim fiyatı	1500
Satış adedi	8
Birim fiyat	0 (veri sonu)
Toplam tutar	????

- 1 ile 100 arasındaki tek sayıları toplayan programı yazınız.

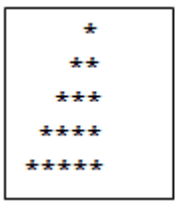
- Aşağıdaki işlemin sonucunu bulan programı yazınız.

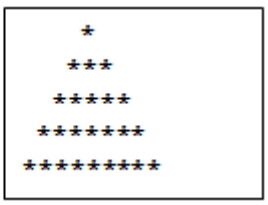
1x2x3x4x5x…..x50

- Aşağıda görüldüğü gibi toplam 10000 YTL elde edebilmemiz için gereken gün sayısını hesaplayınız. İlk gün 1 YTL kazandığımız bir sonraki günde bu değerin iki katı kazandığımız varsayılıyor.

Kazanç	**Toplam kazanç**
1. gün 1	1
2. gün 2(= 2 × 1)	3 (= 1 + 2)
3. gün 4(= 2 × 2)	8 (= 3 + 4)

- Ekrana aşağıdaki şekilde olduğu gibi * karakterini basan programı yazınız.





Aşağıdaki işlem basamaklarını takip ederek belirtilen sorulara ilişkin uygulama faaliyetini yapınız.

İşlem Basamakları	Öneriler
➢ Değişken tablosunu hazırlayınız. ➢ Akış diyagramını çiziniz. ➢ Programı yazınız. ➢ Yazdığınız programı derleyiniz. ➢ Programda hata var ise bunları gideriniz. ➢ Ekran görüntüsünü kontrol ediniz.	➢ Programda kullanacağınız değişkenlerin tipini belirleyiniz. ➢ Değişken isimlendirme kurallarına dikkat ediniz. ➢ Akış diyagramı sembollerinden yararlanınız. ➢ Program satırlarının düzenli olmasına özen gösteriniz. ➢ Matematiksel işlem önceliklerine dikkat ediniz. ➢ Karar ifadelerinin belirlenen şartlara uygun olmasına dikkat ediniz. ➢ Döngü başlangıç, sınır ve artış değerlerine dikkat ediniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Aşağıdaki for ifadelerinden hangisi alt alta 1' den 10 ' a kadar olan sayıları ekrana yazdırmaya yarar?

A)
```
for( say = 1; say <= 10; say ++)
    printf("%d\n", say);
```
B)
```
for( say = 1; say < 10; say ++)
    printf("%d\n", say);
```
C)
```
for( say = 0; say <= 9; say ++)
    printf("%d ", say);
```
D)
```
for( say = 1; say <> 10; say ++)
    printf("%d\n", say);
```

2. Aşağıdaki çıktıyı verecek olan ifade aşağıdakilerden hangisidir? (İp ucu: iç-içe döngüler kullanılacak)

```
1
22
333
4444
55555
```

A)
```
for(a = 1; a <= 5; a = a + 1) {
    for( b = 1; b <= 5; b = b + 1)
    printf("%d", b);
    printf("\n");
    }
```
B)
```
for( a = 1; a <= 5; a = a + 1) {
    for( b = 1; b <= a; b = b + 1)
    printf("%d", a);
    printf("\n");
    }
```
C)
```
for( a = 1; a <= 5; a = a + 1) {
    for( b = a; b <= 5; b = b + 1)
    printf("%d", b);
    printf("\n");
    }
```
D)
```
for( a = 1; a <= 5; a = a + 1) {
    for( b = 1; b < a; b = b + a)
    printf("%d", b);
    printf("\n");
    }
```

3. 10 ile 100 arasındaki sayıların toplamını bularak, bu değeri toplam değişkenine atayan ifade aşağıdakilerden hangisidir? toplam değerinin başlangıç değeri program içerisinde verilecektir.

A) for(a = 10; a <= 100; a = a + 1)
   toplam = toplam + a;
B) for(a = 10; a < 100; a = a + 1, toplam = 0)
   toplam = toplam + a;
C) for(a = 10; a <= 100, toplam = 0; a = a + 1)
   toplam = toplam + a;
D) for(a = 10, toplam = 0; a <= 100; a = a + 1)
   toplam = toplam + a;

4. Aşağıdaki ifadelerden hangisi A-Z arasındaki karakterleri ekrana yazmaya yarar?

A) for(a = 'A'; a < 'Z'; a = a + 1)
   printf("%c", a);
B) for(a = 'a'; a <= 'z'; a = a + 1)
   printf("%c", &a);
C) for(a = 'A'; a <= 'Z'; a = a + 1)
   printf("%c", a);
D) for(a = 'Z'; a <= 'A'; a = a + 1)
   printf("%c", a);

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise "Modül Değerlendirme"ye geçiniz.

MODÜL DEĞERLENDİRME

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet,** kazanamadıklarınız için **Hayır** kutucuklarına (**X**) işareti koyarak kontrol ediniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. İnclude komutu ile istenilen kütüphaneleri belirttiniz mi?		
2. Main fonksiyonu içerisine ana programı yazdınız mı?		
3. Matematiksel işlem yaparken operatörleri kullandınız mı?		
4. Print() fonksiyonu ile verilerin ekranda çıktısını aldınız mı?		
5. Scanf() fonksiyonu ile klavyeden veri okuttunuz mu?		
6. Iki yollu karar yapısını kullandınız mı?		
7. İç içe karar yapısını kullandınız mı?		
8. Çoklu karar yapısını kullandınız mı?		
9. While döngü yapısını kullandınız mı?		
10. For döngü yapısını kullandınız mı?		
11. İç içe döngü yapısını kullandınız mı?		
12. Program yazım işlemlerinin doğruluğunu kontrol ettiniz mi?		

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ–1'İN CEVAP ANAHTARI

1	**C**
2	**A**
3	**C**
4	**A**
5	**D**
6	**B**
7	**B**
8	**A**
9	**C**
10	**C**

ÖĞRENME FAALİYETİ–2'NİN CEVAP ANAHTARI

1	**D**
2	**A**
3	**B**
4	**A**
5	**C**
6	**==, &&, <**
7	**<, ||, <**
8	**>=, &&, <=**
9	**Mektup, 'X', default**

ÖĞRENME FAALİYETİ–3'ÜN CEVAP ANAHTARI

1	**A**
2	**B**
3	**D**
4	**C**

KAYNAKÇA

- ISHIDA Yasuhiro, Hideki MURAKAMI, Ito KOICHI, Gürcan BILDIR , **Bilgisayar Kontrol Teknolojisi,** M.E.B – JICA, Ankara, 2005.